A Grid-enabled Interface to Condor for Interactive Analysis on Handheld and Resource-limited Devices

Ali Arshad[1], Anjum Ashiq[1], Azim Tahir[1], Bunn Julian[3], Ikram Ahsan[1], Newman Harvey[3], Steenberg Conrad[3], Thomas Michael[3], Willers Ian[2]

[1]National University of Sciences and Technology

Rawalpindi, Pakistan

Email: {arshad.ali, ashiq.anjum, ahsan, tahir}@niit.edu.pk

[2]CERN

CH-1211, Geneva 23, Switzerland

Email: Ian.Willers@cern.ch

[3]California Institute of Technology

Pasadena, CA 91125, USA

Email: {conrad,newman}@hep.caltech.edu, Julian.Bunn@caltech.edu, thomas@hep.caltech.edu

ABSTRACT

This paper presents the design and implementation of a Grid-enabled interactive analysis environment for handheld and other resource-limited computing devices. Handheld devices are inherently limited in resources, such as processing power, memory usage, permanent storage and network connectivity. However, handheld devices offer great potential because they provide ubiquitous access to data and round-the-clock connectivity. Our solution aims to provide users of handheld devices the capability to launch heavy computational tasks on computational and data grids, monitor the jobs' status during execution, and retrieve results after job completion. A web service-hosting framework named JClarens has been developed to act as the middleware for job scheduling, submission and execution. Users carry their jobs on their handheld devices in the form of executables (and associated libraries). Files to be analyzed remotely are served by JClarens. JClarens then schedules the jobs on the most suitable farm on the Grid, using monitoring information gathered from all the connected farms. It also manages the transfer of executables and selected files from the handheld devices to the selected farm, the submission of the job to the Condor scheduler on that farm and the subsequent monitoring of the jobs' status. Thus users can transparently view the status of their jobs and finally get back

their outputs without having to know where they are being executed. In this way, our system is able to act as a high-throughput computing environment where devices ranging from powerful desktop machines to small handhelds can employ the power of Grid computing.

Keywords: Grid computing, Handheld Computing, Physics Analysis, Context-aware job submission.

1. INTRODUCTION

Handheld computing and wireless networks hold a great deal of promise in the fields of ubiquitous data access and novel application development. In today's world, mobility in all facets of life is becoming more and more common. People, in general, and the scientific community, in particular, are eager to get hold of information as soon as possible. Mobile computing helps solving the technical communities' desire to get hold of information anywhere and everywhere.

However, the analysis and processing of information to produce useful filtered results is a much bigger challenge, as it requires not just network connectivity, but considerable processing power as well. Most mobile and handheld devices, and even many laptops and desktop machines are extremely limited in the processing power they offer. Many tasks are themselves so CPU-intensive and time-consuming that a single machine seems quite insufficient for them.

Grid computing is poised to become the technology that will eradicate this problem forever. We present here a brief description of a Grid enabled analysis environment for handheld devices that we are developing specially for the high-energy physics (HEP) community. We describe the architecture, design, and implementation of this Grid-enabled system, and then present some of the future developments that we envision for this project.

Handheld devices, considering their present rate of growth, are expected to replace desktop machines in the coming future. This project focuses on leveraging the power of Grid technology to enhance the capabilities of mobile, handheld devices, having the inherent limitations of reduced CPU performance, small secondary storage, heightened battery consumption sensitivity, and unreliable low-bandwidth communication. This enables even small handheld devices to analyze data stored in enormous data stores connected to the Grid.

The CMS (Compact Muon Solenoid) [1] at CERN, going online in 2006, will use such Grid-based data stores for the gigabytes of data it will generate each minute. In its raw form, the data cannot be used to generate any significant results, because of its sheer quantity and complexity. The only way of analyzing this data is by using analysis applications to render it in the form of 2D and 3D diagrams, which scientists can use much more effectively in deriving conclusions about events taking place in the CMS. At the same time, efficient processing of data is required in order to make the analysis process as fast and interactive as possible.

Our aim in this work is to develop a set of physics analysis applications for handhelds and optimize them for maximum performance on the handheld devices. At the same time, we have been developing a Java-based framework, named JClarens, for hosting Web services and Grid services. This framework allows users a single point of access to Grid services, such as data storage services, data replication services, monitoring services and job submission services. Using JClarens, users can search for data on the Grid and launch analysis jobs on the datasets. The problem of assigning jobs to different farms based on monitoring information from those farms is handled by JClarens transparently from the users.

The responsibility of scheduling jobs on a compute farm is currently carried out using the Condor job scheduling system. Condor is a job scheduling and job execution system developed at the University of Wisconsin, Madison. The modular and service-oriented architecture of JClarens helped us to provide an interface for PDAs to submit jobs to Condor. This enables even the resource constrained handhelds to launch lengthy and time-consuming jobs on a large number of input files on a Condor pool, get periodic information on job status, and get back the outputs. In this way, users (or clients) can continue to work conveniently behind a simple interface, while complicated and time-consuming jobs are executed and managed by JClarens on the Grid.

2. RELATED WORK

Mobile computing and Grid computing, if merged, have a great deal of technological potential. This would allow mobile and PDA users to submit jobs on the Grid, and access its tremendous processing power. It would also allow them to access data being generated by the Grid around the clock, and see the results of the analysis on their handheld devices.

Nowadays, a great variety of portable devices are available which includes laptops, PDAs, and mobile phones. This research aims at making the power of Grid computing available to resource-limited devices such as laptops and PDAs, especially the Pocket PC and Palm. At the same time, we have been working on porting popular physics analysis applications to PDAs, because due to the processing power and resources generally required by analysis applications, none of them has yet been ported to the low resource (usually 32MB RAM) and slow processing (typically 200MHz to 400MHz) handheld devices. Moreover, slow, unreliable, and intermittent nature of wireless connections has always been a concern.

For desktop machines, on the other hand, a variety of grid enabled physics analysis applications is available. Some of these include JAS [2], WIRED [3] and ROOT [4]. JAS (Java Analysis Studio) was developed at the Stanford Linear Accelerator Center (SLAC), and is used for 1D and 2D graphical display of histogram data obtained from particle accelerators. Along with graphical display, it also offers various mathematical functions to fit along the displayed data. The user can also view various statistics related to the analyzed data set.

WIRED was developed at CERN in collaboration with SLAC. Wired uses XML based files for graphical rendering of events and sub component geometry information from various particle experiments.

Also developed at CERN, ROOT analyzes special format ROOT files in which data is arranged in a highly efficient, hierarchical structure.

Research is already being done to integrate the two groundbreaking fields of Grid computing and mobile computing [5]. Even Java-enabled mobile phones have been targeted for possible integration with the Grid environment, in order to provide more computationally intensive features on mobile phones [6].

Our Pocket PC based analysis applications are basically built around the Java Analysis Studio (JAS) and WWW Interactive Remote Event Display (WIRED) software, and have been optimized for the Personal Java [7] environment on the Pocket PC 2002. On the server side, JClarens is used to provide a framework on which Grid services are hosted to provide Grid authentication, job submission, job tracking, data access, and file browsing services.

3. ARCHITECTURE OF THE ANALYSIS ENVIRONMENT

The analysis environment consists of two entirely decoupled components. On one side are the resource-limited handheld devices and the applications specifically designed for these devices. On the other end is the JClarens Grid service host, which makes the facilities offered by the Grid available on the handheld devices.

The clients comprise simple, portable applications for handheld and desktop devices, which communicate with JClarens, using SOAP/XML-RPC. They can be programmed in C++, Java, Python or any other language supporting XML-RPC. Once logged on to JClarens, the clients can access the services offered by it. A detailed overview of the services offered by Clarens, and the way its clients communicate with it is described in [8].

The server, as mentioned earlier, basically acts as a Grid service host. It hosts any services and methods that have to be hosted on the Grid. It communicates with clients using the lightweight XML-RPC protocol. This allows clients to be made extremely simple, and abstracts away all the complexity of the hosted Grid services from the client.

To provide Grid functionality on the handheld device, a separate service for job submission has been implemented in JClarens. This service (hosted on the resource broker) receives job submission requests from clients, and then tries to find out the most suitable farm available for job submission, using monitoring information received from all the connected farms (or standalone computers). Once this has been located, a job ID is assigned to that particular job, the job submission request is forwarded to that particular farm, and a record of where the request was forwarded is stored in a database. Any subsequent requests for checking the job status, killing the job or retrieving its outputs are forwarded to the JClarens server on the farm concerned.

Once a job request is forwarded to a particular farm, the JClarens server on that farm creates a temporary staging directory for Condor on the server machine. The executable, required libraries, a dynamically generated submit file and input files are copied into the staging directories, and "condor_submit" is called to submit the job to the Condor pool on the farm.

The clients can subsequently check the status of their jobs (which can be running on any one of the available farms) at any time, without having to know where the jobs are actually being executed. Once the jobs are complete, the user can retrieve the results of his jobs at any time, using the relevant XML-RPC calls.

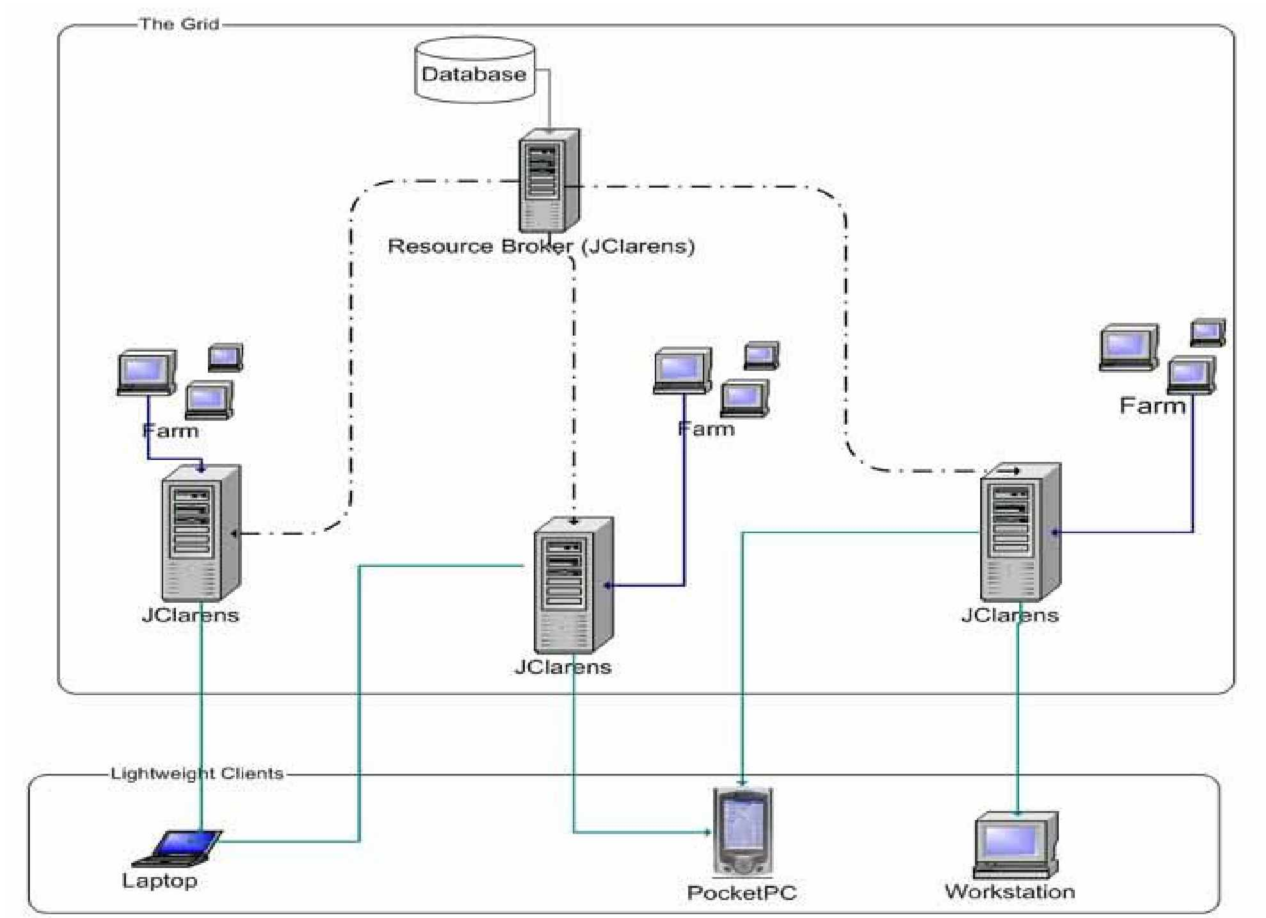

Fig. 1. A general architectural overview of the analysis environment

4. SERVER-SIDE DESIGN

The server side centers mainly on the JClarens framework and the services provided by it. A detailed description of the clients is given in section 5. Here we describe the overall design of the server-side Grid service host, how jobs are actually submitted to the Condor job scheduling system, and how they are managed. The major components on the server side are as follows:

4.1. XML-RPC server

A modified version of Apache's XML-RPC classes is used to provide XML-RPC parsing and processing capabilities. The XML-RPC server, encapsulated in a Java servlet, is responsible for processing incoming requests, extracting information about what services and methods to invoke, and writing out responses in XML. In addition, we have also modified it to be able to write back responses directly in binary form, to reduce overheads during binary file transfers.

4.2. Authentication and authorization services

Authentication and authorization of valid users and virtual organizations (VOs) is one of the most important requirements of any Grid-enabled system. JClarens carries out authentication using a GSI-based [9] security protocol. The server and clients use X.509 certificates and RSA keys for authentication. A detailed description of JClarens authentication protocol is given in [10].

Authorization is carried out using access control lists (ACLs) maintained in a database. The database system available by default is MySQL, although any other DBMS can be easily plugged-in by providing a suitable JDBC driver, and modifying configuration settings in a properties file.

The database contains the distinguished names (DNs) or substrings of DNs that are allowed or denied access to various services. A set of methods is also available for manipulating these ACLs by system admins. Besides users' DNs, the names of the VOs can also be used to configure access control for large groups of users or VOs.

4.3. Data access services

JClarens also provides data browsing, searching and downloading capabilities. Using the "file" service, users can browse files, search for files using "wildcards", and search within files. They can also download files and find out the validity of downloaded files using md5 hash values.

4.4. Monitoring service

A prototype monitoring service has been developed which reports monitoring information to a central JClarens server, hereafter known as the resource broker. The monitoring service gathers monitoring information using a companion End-host Monitoring Agent (EMA), a monitoring product also being developed in NUST. EMA gathers information such as the CPU clock rate, CPU usage, Memory Usage and Last 1, 5 and 15-minute average load values. These parameters are used to calculate a "load coefficient", which describes the load on a particular system in quantitative terms. At the moment, the coefficient is calculated using an empirical formula:

$$Coeff. = ((1-CPU\_Usage/100)*-1*Clock\_Rate) + \Sigma\, a_i M_i$$

Where i = number of monitoring parameters being used

M_i = the value of the i'th monitoring parameter

and a_i = an experimentally determined weighting factor for the i'th monitoring parameter

Currently the monitoring parameters being used are the percentage of memory being used, the Disk IO in Mbps, the average CPU load during the past one minute (Load1), and the number of currently executing processes.

Every few seconds (10 by default), this coefficient value is sent to the resource broker as an XML-RPC call, where it is stored in a database, along with the URL of that particular server. This monitoring information is used at the time of job submission to decide where to forward jobs for execution.

4.5. Job submission service

The job submission service is designed to submit jobs to the least loaded computer or farm on the network (LAN or WAN). The job submission service receives requests for submission of a job from clients with the following parameters:

1) The name of the job
2) The binary code of the executable
3) The submit file
4) The name of the submit file
5) The names of the input files

The clients themselves do not provide the input files at present. Instead the clients request the JClarens server for files available in its file publishing area and select the ones on which they wish to do analysis.

On receiving a job submission request (condorSubmit), the receiving JClarens server gets the minimum load coefficient reported by the other peer servers from the resource broker. It then forwards the job to the least loaded peer. Note that the peer is basically a JClarens server running on another computer or on the head node of a compute farm.

When a peer JClarens server receives a forwarded job submission request from another peer, it creates a temporary staging directory for Condor on the local hard disk. It then generates a unique job ID for the request and creates a subdirectory in that folder with the same name as the job ID. A number of subdirectories are then created in the job's particular subdirectory equal to the number of input files and the executable and the submit file are copied into each of the subdirectories naming them the same as the file names in the request. If the forwarding peer is the local host itself, the server copies the file from the local file system to the appropriate subdirectory directly. Otherwise, it downloads the file from the forwarding JClarens server using the "file.read" call.

Once the staging directories are prepared, JClarens submits the jobs to Condor by executing "condor_submit" on the submit file. Condor submits jobs in logical groups called "clusters". A mapping between the jobs themselves and the clusters to which the jobs have been submitted is also stored in a database.

Condor [11] then takes over the allocation of the individual jobs to different machines in the Condor pool, and ensures that the jobs are completed and the results returned to the respective directories. While Condor executes the job on the pool, the client can repeatedly poll JClarens to check the status of his jobs.

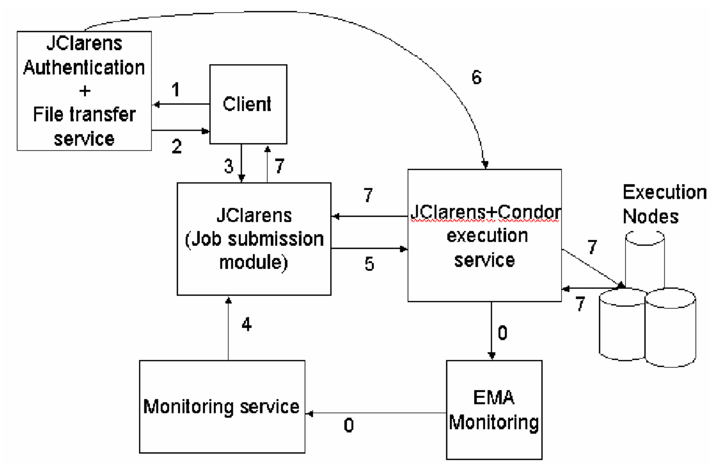

Fig. 2. An overview of the steps involved in submitting jobs and checking their status:

0. EMA periodically sends monitoring info to JClarens monitoring interface
1. User authenticates, asks for list of files, browses files
2. User gets back list of files, selects files, downloads files if required
3. User submits jobs
4. Monitoring info is read by JClarens to select best place to submit job
5. Job is forwarded to best available site
6. Input Files are downloaded to remote execution site, if necessary
7. User checks status of jobs repeatedly, gets back outputs when desired etc.

In case of successful submission, JClarens returns a unique ID generated for the job, and this is sent to the initial submitting client through the master JClarens. A table mapping the job ID with the peer where the job has been actually submitted is also updated. After that, the client uses this job ID to inspect the status of the job, or to retrieve its status at any later stage.

JClarens retrieves the status of a job with a particular ID by finding out the peer where the job with that ID has been submitted. The execution of "condor_q" on that peer gives the status of all the jobs running on the machine. The status of the jobs for that particular job ID is extracted by filtering out the clusters mapped to that job ID from the output of "condor_q".

Once the job is complete, the client can retrieve the outputs of all his jobs through JClarens. Clients can look at all the output files produced during execution, and download the desired output files. In this way, a complex and time-consuming set of jobs can be rapidly executed on an underutilized Condor pool, and its results displayed and visualized on the submitting client.

For fault tolerance and high availability, the system has been designed to be entirely self-healing. Whenever a new JClarens server comes up and reports to the resource broker, the resource broker informs all the available servers of this new server. In this way, if at any stage, the central JClarens resource broker goes down, one of the other servers detects this faulty condition automatically, and can proclaim itself as the central resource broker to all the other available servers. In this way, it is ensured that the system is able to heal itself and remain available for the clients, even if the central resource broker becomes inaccessible.

Using this architecture, even users on slow-processing PDAs connected to wireless networks can accomplish complex jobs. They can carry out remote processing of lengthy operations on powerful machines in the Condor pool. This allows even handheld devices to exploit the power of Grid computing.

5. GRID-ENABLED CLIENTS

The capability and utility of the system has been validated using clients that have been developed for the Personal Java runtime environment on iPAQ Pocket PCs. A description of the clients developed and the features offered by them for interactive analysis follows.

5.1. Java Analysis Studio (JAS) and JASONPDA

Java Analysis Studio (JAS), as stated earlier, is being developed at Stanford Linear Accelerator (SLAC). JAS is a physics analysis tool used for analyzing data obtained from linear accelerators in the form of 1D-2D histograms. Apart from 1D-2D analysis, JAS offers numerous other facilities, which include comparison of displayed histograms with predefined mathematical functions (Quadratic, Cubic, Gaussian, polynomial, Lorentzian etc), the fitting of these functions over the displayed histograms for statistical analysis (peak value, average value etc), and executing your own analysis code on selected datasets.

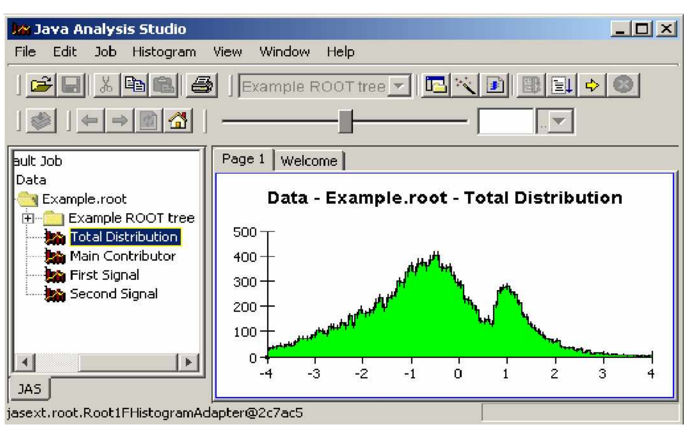

Fig. 3. Java Analysis Studio (JAS) (above) running on a Desktop Machine.

JASOnPDA was our first application for PocketPC. JASOnPDA is the scaled down version of Java Analysis

Studio, especially designed for constrained handheld devices. JASOnPDA provides essential analysis utilities of Java Analysis Studio on PocketPC devices, and was developed using J2SE 1.1. JASOnPDA allows mobile users to log on to the JClarens server using a certificate-based authentication procedure. Once successfully authenticated, the user is allowed to access files stored at the server. The remote browsing facility allows users to browse the directories served by JClarens and look for desired ROOT files. The selected ROOT file is analyzed and a tree structure displays the hierarchy of objects in the ROOT file. The user can move along the tree, selecting any object from the tree structure, and the selected object will be displayed in the form of 1D-2D Histograms in the display panel.

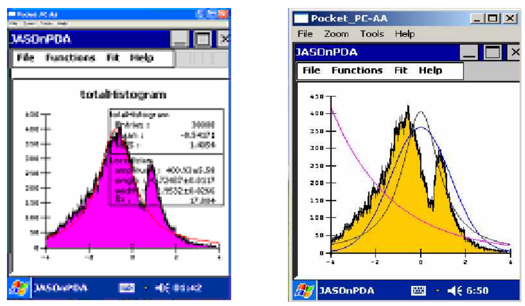

Fig. 4. JASOnPDA running on a PDA showing its features of histogram plotting, function fitting, and statistics calculation.

In order to submit jobs to the Grid using JClarens, JASOnPDA allows users to select the file that will be used as the job's executable. It also allows users to type in or select a file that will be used as the submit file. Once this is done, the user can select the input files that will be used as the input for the jobs, and finally submit the job. A notification is received when the job is successfully submitted, after which JASOnPDA periodically polls JClarens for the jobs' status. As soon as the job is complete, a menu is displayed showing the files on which the job execution was successful. The user can then select a particular file, to get the resulting output of the job execution on that particular file. In this way, users on handheld devices can submit jobs on powerful compute farms, and at the same time, get back outputs of the job executions.

Statistical fitting features from JAS have also been ported. The user can also view statistics information related to the histogram displayed on the screen. Keeping in view the small screen size of the PocketPC, different viewing options are also provided.

JASOnPDA had some efficiency issues with the very first release. The time taken to download the files onto the PocketPC and display the histograms was much more than the time taken on desktop machines. Scrolling and other display functions were also quite slow. These issues have mostly been resolved in later versions, and the result is a much better-performing analysis application.

To overcome the issue of intermittent, unreliable connections during the transfer of large datasets and files, the downloading process for large files is carried out by dividing the size of the file into smaller chunks, and downloading those smaller file chunks in steps, rather than using a single connection to transfer the entire file. This allows us to check point the file transfer process and ensure that if there is a disconnection at any stage, the entire dataset is not lost, and downloading can be resumed from the latest check point.

Recently, more features for extensibility have been added as well. Interfaces have been exposed that allow users to write their own analysis classes for different file formats. This allows users the flexibility to select the procedure by which their file will be analyzed and its contents displayed. To specify the class to be used for analysis, the user only has to give the name of the class in a simple properties file. In this way, the user can easily plug in the classes that he wants to use for handling new file formats and user-specific custom file formats.

5.2. WWW Interactive Remote Event Display (WIRED) and WIREDONPDA

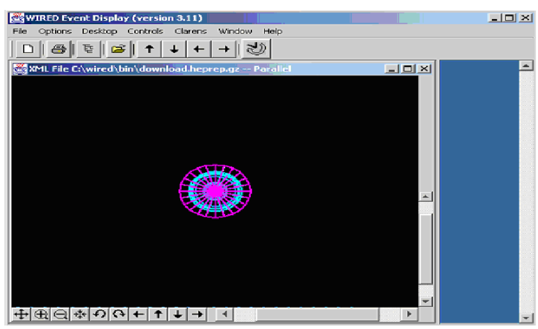

Fig. 5. WWW Interactive Remote Event Display running on a Desktop machine.

WWW Interactive Remote Event Display (WIRED) was a joint venture of Stanford Linear Accelerator (SLAC) and European Organization for Nuclear Research (CERN). WIRED is one of the first Event Displays written in Java for use on the World-Wide-Web. It provides a framework for writing event displays. It is in active use by the BaBar and GLAST experiments and the LCD detector study at SLAC.

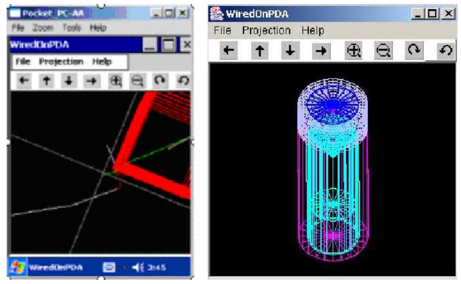

Fig. 6. Two views of WiredOnPDA displaying events from a HepRep2 file (left) and detector geometry (right).

WiredOnPDA is another of our analysis applications developed for PocketPC devices. As the name suggests, WiredOnPDA provides analysis features of WWW Interactive Remote Event Display (WIRED) on PocketPC. WiredOnPDA accesses data using JClarens in the same way as JASOnPDA.

As mentioned earlier, the user has to pass a security check by providing a valid certificate and key. Once authenticated, the user can access the remote server and can select any HepRep2 event file placed on the server.

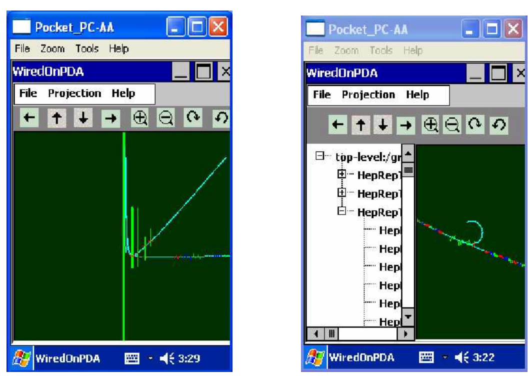

Fig. 7. Two views of WiredOnPDA displaying event data and the structure of a HepRep2 file in separate panes.

Once the user selects a file from the JClarens server, the file is downloaded into the RAM and a SAX XML parser parses the information stored in the file. “Drawables” are then extracted from the parsed data and are displayed in a hierarchical tree structure in WiredOnPDA tree panel. User can then select any “drawable” from the tree and it will be displayed in the display console. Again, keeping in mind the small screen size of the PocketPC, various display options are provided in order to utilize maximum screen space for event display. As shown in the figures, the application is provided with a tool bar that allows the user to scale, rotate or zoom the displayed event for better analysis.

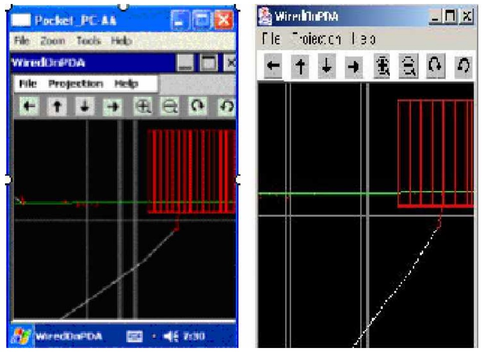

Fig. 8. Two views of WiredOnPDA (at right on a PDA, and at left on a PC) displaying the same event as figure 5 but from a different angle.

WiredOnPDA also had some issues of concern with initial release, most of which were regarding performance. The main reason for inefficient performance was the poor parsing speed of SAX parsers in PersonalJava. Performance analysis has shown that the reason for this slow parsing is because of the differences in implementation of PersonalJava virtual machine compared to J2SE virtual machines. To address this issue various parsers were tried and tested which included Xerces, Crimson, KXML and Piccolo. Piccolo has so far proved to be the fastest performer, with the best possible results out of all the other parsers.

6. FUTURE WORK

There are still several ideas that are yet to be implemented in this project. The most important of this is an even further decentralized architecture, where there is no master server at all. Instead all the servers are true peers of each other and can get monitoring information from a decentralized source.

A self-organizing neural network (SONN) can be introduced to learn usage and load patterns and thus schedule jobs more efficiently.

Another feature that is planned to be added is the ability to keep a catalog of all files available in all the servers on the network, thereby allowing users to discover any dataset they require without prior knowledge of the server holding that dataset.

A major advance would be the ability to get job outputs back as soon as they are produced rather than having to retrieve them when the entire job is complete. This would be a major advance towards more interactivity.

Further work on the client side includes the extension of the analysis environment to run on PocketPC Phone Edition devices and to be able to use fast cellular networks such as GPRS, for data communication with the remote data servers, over the Internet.

7. CONCLUSION

The analysis environment has been developed to contribute to the scientist and physicist community all around the world, to help them in their quest for ubiquitous access to data, over wired and wireless links. But it is yet to reach the high performance standards that are so easily achieved in case of desktop PC-based applications. This is clearly indicated by some of the performance comparisons we presented earlier. The slow performance of the handheld devices is the major barrier that has to be overcome to achieve true interactivity. However, the use of a distributed job execution and data analysis environment has enabled us to speed up our analysis applications to a great extent. Even if a set of jobs is submitted on a standalone analysis server, users can get back the results of analysis on almost fifteen event files (each event stored in a 1MB file) in less than a minute, whereas a standalone PocketPC could not process more than three or four files per minute.

Our PocketPC based applications were demonstrated in ITU Telecom World 2003 as a part of the "GRID-Enabled Physics Analysis" demonstration. Also JASOnPDA was presented at the first Grid Analysis Environment (GAE) workshop at Caltech in June, and attracted a great deal of attention, due to the fact that it was the first physics analysis application ported to the PDA.

WiredOnPDA is expected to gain at least the same amount of attention as its predecessor. Our work on this analysis environment, however, is far from over. Already our current work proves that resource-constrained devices such as the PocketPC can be interfaced with the Grid, and can play a vital role in the realization of the idea of a Grid-Enabled Analysis Environment (GAE). Mobile and ubiquitous computing has yet to reach a mature level, where it can compete with desktop PCs in the kind of applications it can offer. The achievement of the above mentioned goals would prove to be a giant leap towards the attainment of this level of maturity.

8. REFERENCES

[1] Compact Muon Solenoid Outreach Activities (http://cmsinfo.cern.ch/)

[2] Java Analysis Studio (http://jas.freehep.org/)

[3] WWW Interactive Remote Event Display (http://wired.freehep.org/)

[4] The ROOT System (http://root.cern.ch)

[5] Thomas Phan, Lloyd Huang, Chris Dulan, "Challenge: Integrating Mobile Wireless Devices Into the Computational Grid", Proceedings of the 8th annual international conference on Mobile computing

and networking. September 2002, pp 271-278.

[6] Miika Tuisku, “Wireless Java-enabled MIDP devices as peers in Grid infrastructure” (http://grid.cesga.es/eabstracts/wireless_grid.pdf)

[7] Sun Microsystems Inc., Personal Profile, Personal Basis Profile, and Personal Java (http://wireless.java.sun.com/personal/)

[8] Conrad D. Steenberg and Eric Aslakson, Julian J. Bunn, Harvey B. Newman, Michael Thomas, Frank van Lingen,”Clarens Client and Server Applications” in Conference on High Energy Physics (CHEP), 2003, La Jolla, California, paper no. MONT008.

[9] V. Welch, F.Siebenlist, I. Foster, J. Bresnahan, K.Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke, “Security for Grid Services” in Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press, June 2003, pp.48-57.

[10] Conrad D. Steenberg and Eric Aslakson, Julian J. Bunn, Harvey B. Newman, Michael Thomas, Frank van Lingen, “The Clarens Web Services Architecture” in Conference on High Energy Physics (CHEP), 2003, La Jolla, California, paper no. TUCT005.

[11] Michael Litzkow, Miron Livny, and Matt Mutka, "Condor - A Hunter of Idle Workstations", Proceedings of the 8th International Conference of Distributed Computing Systems, June, 1988, pp. 104-111.